BAB III

PELAKSANAAN KERJA MAGANG

3.1. Kedudukan dan Koordinasi

Kerja magang yang dilakukan di PT. XYZ sebagai *data engineering intern* berada di ranah departemen *tech*, divisi *data*. Didalam divisi *data*, total tim berjumlah 3 (tiga) orang dengan komposisi 1 (satu) orang sebagai *lead* dan 2 (dua) orang lainnya sebagai *data engineering intern*. Komposisi tim tersebut dapat dibagi sebagai berikut:

Data Lead (Senior Business Intelligence Engineer) : Alief Aziz

Data Engineer Intern 1 : Steven Johan

Data Engineer Intern 2 : Alexander Jordan

Dalam koordinasi pekerjaan sehari-hari, *data engineer intern* bertanggung jawab ke Bapak Alief Aziz selaku *lead* dan pembimbing lapangan dari kerja magang. Pemagang juga bekerja bersama dengan *data engineer intern 2* yang bernama Alexander Jordan dalam menjalankan fungsi dari divisi data. Alexander Jordan merupakan mahasiswa yang datang dari universitas lain dan memiliki periode serta status kerja magang yang sama dengan pemagang. Dalam pembagian tugas sehari-hari di antara 2 orang *data engineer intern, data lead* memberikan jenis pekerjaan dan beban kerja yang sama.

Pada awalnya, semua koordinasi dari divisi luar data dilakukan melalui *lead*. Setelah menggali *requirement* dan menentukan ekspektasi output yang diharapkan

dari stakeholders mengenai suatu pekerjaan, maka *lead* akan mendelegasikan pekerjaan tersebut ke *data engineer intern* sesuai dengan beban kerja yang sedang dijalankan. Setelah 2 (dua) bulan, *data lead* mulai membiasakan *data engineer intern* untuk langsung berkomunikasi dengan *stakeholders* di *virtual* meeting untuk melakukan eksplorasi masalah dan memberikan solusi. Pada tahap selanjutnya, *data lead* memberikan kebebasan kepada *data engineer intern* untuk langsung menghubungi *stakeholders* apabila ingin membahas mengenai eksplorasi masalah dan ekspektasi *output* yang diharapkan. Sebaliknya, *stakeholders* juga disarankan untuk menghubungi *data engineer intern* secara langsung jika memiliki *request*. *Stakeholders* dapat datang dari divisi antar departemen tech maupun divisi dari luar departemen *tech.* Semua interaksi yang terjadi antara *data engineer intern* dengan *stakeholders* akan dilaporkan kembali kepada *lead* melalui *daily standup meeting*. Dalam melakukan pekerjaan sehari-hari, *lead* memperbolehkan *data engineer intern* untuk menghubungi *lead* melalui aplikasi *Slack* apabila terdapat kendala atau masalah yang tidak dapat diselesaikan.

3.2. Tugas yang dilakukan

Pekerjaan yang dilakukan *data engineer intern* di PT. XYZ datang dari 2 (dua) sumber, yaitu *stakeholders* dan internal dari *data lead.* Pekerjaan / *request* / *task* yang diterima dari *stakeholders* bersifat unik namun repetitif dengan tingkat kompleksitas yang berbeda-beda. Pekerjaan yang datang dari *data lead* untuk internal divisi biasanya merupakan pekerjaan yang bersifat optimisasi dan

monitoring. Tugas yang dilaksanakan *data engineer intern* saat kerja magang dapat dikelompokkan sebagai berikut:

1. Melakukan *development, optimization, dan testing* kode ETL (*Extract, Transform, Load)* data pada sistem *automatic scheduler Airflow* yang menggunakan bahasa *Python* dan *CRON Job*.
2. Melakukan operasi CRUD *(Create, Read, Update, Delete)* pada *database MySQL* dan *GCP (Google Cloud Platform).*
3. Melakukan investigasi dan *clear-out* data yang tidak benar pada sumber data seperti *MySQL* dan *Google Spreadsheets* untuk menjaga kualitas data.
4. Bekerja sama dengan *stakeholders* dalam pembuatan dan optimisasi *Google Spreadsheets* yang umum digunakan karyawan PT. XYZ. Hal yang dilakukan dapat berupa pembuatan dan optimisasi *formula* maupun *dashboard* sesuai kebutuhan *stakeholders,* investigasi dan melakukan *hotfix* terhadap masalah yang terkait sumber data, dan lain sebagainya.
5. Melakukan dokumentasi definisi data yang datang dari berbagai sumber dan melakukan *data sync* dengan divisi *external* terkait perubahan maupun penambahan di definisi data.

3.3. Uraian Pelaksanaan Kerja Magang

Proses kerja magang dilakukan dengan durasi total 25 minggu. *Timeline* aktivitas kerja magang dari minggu pertama hingga minggu kelima dapat dilihat di Tabel 3.1.

Tabel 3.1 Tabel Timeline Aktivitas Kerja Magang Minggu Pertama hingga Minggu Kelima

No	Aktivitas	Minggu ke-				
		1	2	3	4	5
1	Mempelajari dan melakukan instalasi *software* yang akan digunakan					
2	memahami informasi-informasi fundamental mengenai PT. XYZ dan sistem koordinasi tim data dengan *stakeholders*					
3	Melakukan data *clearout* pada tabel *quotation_items* di *database MySQL*					
4	Menambahkan dan mendokumentasikan kolom *creator* ke *spreadsheet transitem data dump*					
5	*Populate timestamp* data ke table *taggables* di *MySQL*					
6	Membuat dan mendokumentasikan *spreadsheet ads data dump*					
7	Melakukan *update* data *commercial_category* di *MySQL*					
8	Menambahkan dan mendokumentasikan kolom *reason* ke *spreadsheet quoted data dump*					
9	Melakukan *update sales parameter* di *spreadsheets*					
10	Melakukan *update* data *tags* di *MySQL*					
11	Membuat *spreadsheet pricing engine lem data dump*					

3.3.1. Mempelajari dan Melakukan Instalasi Software yang Akan Digunakan

Infrastruktur utama yang digunakan di PT. XYZ adalah *scheduler software* bernama *Airflow*. *Airflow* digunakan untuk menjalankan kode berbentuk *py (Python)* dengan urutan yang telah ditentukan sebelumnya di file berformat *dags*. *Airflow* hanya dapat dijalankan di OS berbasis *Unix /*

Linux sehingga WSL (*Windows Subsytem for Linux)* akan digunakan untuk tetap menjalankan semua operasi dari OS *Windows*.

Kode tersebut akan di-*edit* menggunakan *text editor* yang cukup popular dan canggih yaitu *Microsoft Visual Studio Code*. Dikarenakan semua kode yang ada di PT. XYZ tersimpan di *remote Git*, maka *extension* yang ada di *Visual Studio Code* akan digunakan sebagai *client*. Selanjutnya, semua data PT. XYZ disimpan di *database MySQL*, untuk mengakses dan berinteraksi dengan data-data tersebut akan digunakan *software* bernama *MySQL Workbench*. *Software* yang digunakan untuk komunikasi adalah *Slack*. Selain hal yang telah disebutkan, *software-software* lain bersifat *web-based* seperti *Google Sheets*, *Jira* untuk *sprint planning*, dan lain sebagainya sehingga hanya perlu browser untuk mengakses.

3.3.2. Memahami Informasi-Informasi Fundamental Mengenai PT. XYZ dan Sistem Koordinasi Tim Data Dengan Stakeholders

Data engineer intern diminta untuk mempelajari informasi fundamental PT. XYZ seperti struktur koordinasi, kultur perusahaan, dokumen administrasi yang akan diproses setiap bulan dan cara memprosesnya, dan lain sebagainya.

3.3.3. Melakukan Data Clearout pada Tabel Quotation_Items di Database MySQL

Isu yang dihadapi oleh *stakeholders* adalah terdapat data *quotation_items* yang seharusnya sudah dihapus dari *database* tetapi tetap muncul ketika diinvestigasi.

id	code	product_name	specifications	quantity	unit	...
1	Q-20180914g2g7wx	Kartu	{keterangan":"Flyer per Nama @26 nama"}"	300	pcs	...
2	Q-2018091436Fsh9	Stempel	{keterangan":"Bahan : Flexy"}"	500	pcs	...
3	Q-2018091410yI7D	Pulpen	{keterangan":"ukuran 60x160"}"	400	pcs	...
4	Q-2018091447vE6t	Other	{keterangan":"1/2 folio (21x16cm)	100	pcs	...
5	Q-2018091453Ry6n	Stiker	{keterangan":"Ukuran 8x8	250	pcs	...

Gambar 3.1 Contoh Data *Quotation_Items*

Pada Gambar 3.1, terdapat contoh data *quotation_items*. Pada tahap ini, mahasiswa diminta untuk bekerja sama dengan *stakeholders* terkait untuk mengidentifikasi *id* dari *quotation_items* yang seharusnya sudah tidak muncul. Setelah proses identifikasi dilakukan, maka akan dibuat *MySQL Script* untuk menghapus data terkait yang dapat dilihat pada Gambar 3.2.

```
DELETE * FROM quotation_items WHERE id=1;
DELETE * FROM quotation_items WHERE id=4;
```

Gambar 3.2 *Script Delete* Data di Tabel *Quotation_Items*

3.3.4. Menambahkan dan Mendokumentasikan Kolom Creator ke Spreadsheet Transitem Data Dump

Pada kasus ini, *stakeholders* membutuhkan informasi baru di *transitem data dump spreadsheet*. *Transitem data dump* merupakan hasil ETL yang disimpan di dalam *Google Sheets* sehingga dapat diakses dengan mudah oleh siapa saja. Urutan dan nama kolom yang akan di *load* ketika kode

terpanggil, dapat dikontrol dari *file* dengan nama *transitem_a63t_spsheet_filter_col_a24.py*. Mengingat kolom *creator* yang sudah ada di *unfiltered data dump*, tidak diperlukan untuk memodifikasi kode yang memproses data dari *extract* dan *transform*. Kolom dapat ditambahkan dengan dimasukkan ke *list* kolom yang akan di-*load* seperti pada Gambar 3.3.

Gambar 3.3 Menambahkan Kolom *Creator* ke *Load List*

Penambahan kolom *creator* akan didokumentasikan di *spreadsheet data definition* untuk memastikan informasi kolom *transitem data dump* tetap *up-to-date*.

3.3.5. Populate Timestamp Data ke Table Taggables di MySQL

Tabel *taggables* menyimpan *datetime* dari perubahan status suatu *quotation_items* atau *transaction_items*. Struktur dari tabel *taggables* dapat dilihat di Gambar 3.4.

tag_id	taggable_id	taggable_type	created_at	updated_at

Gambar 3.4 Struktur Tabel *Taggables*

Data *timestamp* yang telah disediakan oleh *stakeholders*, dapat langsung dimasukkan ke tabel *taggables* menggunakan *script INSERT* di *MySQL*.

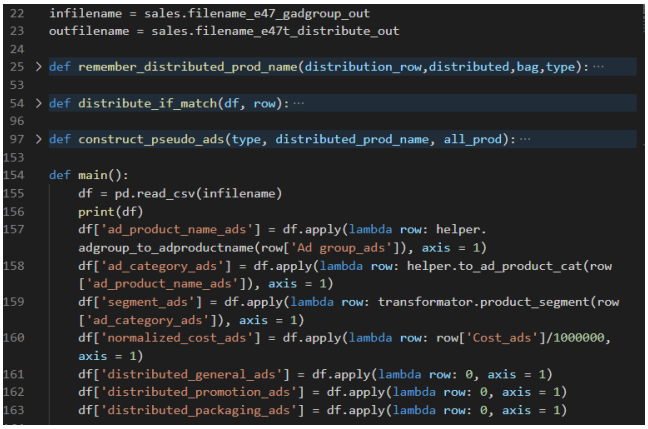
3.3.6. Membuat dan Mendokumentasikan Spreadsheet Ads Data Dump

Proses pembuatan *ads data dump spreadsheet*, dimulai dari tahap *Extract* terlebih dahulu, dimana data diambil langsung dari *database* PT. XYZ yang dapat dilihat di Gambar 3.5.

```
def main():
    # output style, in case for debug
    # display = pd.options.display
    # display.max_rows= None
    # display.max_columns= None
    # display.max_colwidth= 199
    # display.width= None

    print('filename: '+outfilename)
    df = extract()
    df.to_csv(outfilename, index=False)
    print('written to '+outfilename)

def extract():
    print("--- retrieving ads ---")
    start_time = time.time()
    ads_df = cache.get_ads()
    ads_df.columns = ads_df.columns.map(lambda x: str(x)+'_ads')
    print(ads_df.shape)
    print("--- ads: %s seconds ---" % (time.time() - start_time))
    return ads_df

if __name__ == '__main__':
    # Initialize client object.
    main()
```

Gambar 3.5 Proses *Extract* Data *Ads* dari *Database*

Pada Gambar 3.5, data *Ads* diambil dari *database* untuk dilanjutkan ke tahap *Transform*. Pada tahap *Transform*, *raw* data yang diambil diproses sesuai dengan data yang diinginkan oleh *stakeholders* yang dapat dilihat di Gambar 3.6.



Gambar 3.6 Proses *Transformation* Data *Ads*

Setelah melakukan proses *Extract*, semua data akan di-*load* ke *spreadsheet ads data dump.* Kode *load* dapat dilihat pada Gambar 3.7.

```
dag_id='summarize_spsheet_sales_ads'
# daily at 0,3,5,8,11,17 UTC or 7,10,12,15,18,24 GMT+7
with DAG(dag_id, start_date=datetime(2020,3,21), schedule_interval="0 0,3,5,8,
11,17 * * *", default_args=DAG_DEFAULT_FILES, catchup=False) as dag:
    e47 = PythonOperator(task_id="gadgroup_e47", python_callable=gadgroup_e47.
    main)
    e47t = PythonOperator(task_id="gadgroup_e47t_distribute_a48",
    python_callable=gadgroup_e47t_distribute_a48.main)

    e47 >> e47t
```

Gambar 3.7 Melakukan *Load* Data *Ads* ke Spreadsheet

3.3.7. Melakukan Update Data Commercial_Category di MySQL

Pada *requests* ini, *stakeholders* ingin mengubah data kolom *commercial_category* pada tabel *transaction_items*. Data di kolom ini seharusnya hanya mengandung jenis kategori suatu *transaction_items* seperti *new-wave, enterprise,* dan sebagainya. Akan tetapi, setelah dilakukan investigasi, ditemukan bahwa terdapat beberapa *transaction_items* yang *commercial_category*-nya diisi dengan nilai angka. *Requests* ini diselesaikan

dengan cara melakukan *update* data dari angka menjadi *null* menggunakan *script MySQL*.

3.3.8. Menambahkan dan Mendokumentasikan Kolom Reason ke Spreadsheet Quoted Data Dump

Kolom *reason* dapat ditambahkan dengan memasukkan nama kolom *reason* ke daftar kolom yang akan di-*push* ke *spreadsheet quoted data dump*. Dikarenakan data sudah ada di *transformed* data, maka hanya tinggal perlu memodifikasi *filter* kolom saja.

3.3.9. Melakukan Update Sales Parameter di Spreadsheets

Pada kasus ini, *spreadsheet sales commision scheme* dipakai sebagai data input untuk proses *transform* lebih lanjut. Perubahan yang diinginkan oleh *stakeholders* adalah mengubah nilai insentif dari *associate_new_wave*. Hal ini dapat dilakukan dengan langsung mengubah *cell incentive* ke nilai yang diinginkan. Contoh data dari *sales commision scheme* dapat dilihat di Gambar 3.8.

role	period_start	period_end	target	treshold_type	treshold	tresl	incentive_type	incentive(%)
associate_new_wave	2020-04-01 00:00:00	2021-06-30 23:59:59	##########	value	########		normal	2.00

Gambar 3.8 Contoh Data *Sales Commision Scheme*

3.3.10. Melakukan Update Data Tags di MySQL

Tabel *tags* merupakan *master* tabel yang menyimpan informasi mengenai *id* sebuah tag dan informasi pendukung lainnya. Perubahan yang

diinginkan oleh *stakeholders* dapat dicapai dengan melakukan *update* menggunakan *MySQL Script*. Struktur dari tabel *tags* dapat dilihat di Gambar 3.9.

id	label	description	created_at	updated_at
1	quick-quote	Kuotasi yang baru datang dengan waktu produksi terdefinisi oleh customer	NULL	NULL

Gambar 3.9 Struktur dan Contoh Data di Tabel *Tags*

3.3.11. Membuat Spreadsheet Pricing Engine Lem Data Dump

Pada kasus ini, *stakeholders* membutuhkan data *pricing engine* tipe lem untuk dianalasis lebih lanjut. *Pricing Engine* merupakan *tools* yang digunakan untuk mendapatkan harga termurah bagi suatu barang, sumber dari data *Pricing Engine* berasal dari *supplier* dan disimpan dalam bentuk JSON.

```
44  def extract_lem_vendor_df_from_json(lems_vendor_obj):
45      # Result object
46      lem_vendor_df = pd.DataFrame()
47
48      # Level 1 Table (lem Table)
49      str_lem_vendor_obj = {}
50      str_lem_vendor_obj[ 'title' ] = []
51      str_lem_vendor_obj[ 'modified_at' ] = []
52      str_lem_vendor_obj[ 'vendor_id' ] = []
53      str_lem_vendor_obj[ 'tipe' ] = []
54      str_lem_vendor_obj[ 'ukuran_panjang' ] = []
55      str_lem_vendor_obj[ 'ukuran_lebar' ] = []
56      str_lem_vendor_obj[ 'harga_lem' ] = []
57      str_lem_vendor_obj[ 'biaya' ] = []
58      str_lem_vendor_obj[ 'biaya_minimum' ] = []
59      str_lem_vendor_obj[ 'biaya_final' ] = []
60
61      for lem_vendor_obj in lems_vendor_obj:
62          # LOGGING
63          # print( json.dumps(calc_history, indent=2) )
64          # with open( "a.temp", "w" ) as outfile :
65          #     json.dump( calc_history, outfile )
66          # outfile.closed
67
68          # FILL LEVEL 1 DATA
69          str_lem_vendor_obj = fill_lem_vendor_data( str_lem_vendor_obj,
            lem_vendor_obj )
70
71      lem_df = pd.DataFrame( str_lem_vendor_obj )
72      print( lem_df )
```

Gambar 3.10 Proses *Extract* Data dari *File* JSON

Pada Gambar 3.10, Kode *extract* mengambil informasi file JSON yang menyimpan data *pricing engine* – lem. Dikarenakan tipe file yang berbeda, proses *extract* berbeda dengan cara konvensional biasanya. Setelah melalui proses ekstraksi, data akan dilakukan transformasi sesuai dengan kebutuhan *stakeholders* dan akan di-*push* ke *spreadsheet pricing engine – lem* untuk digunakan.

Timeline aktivitas kerja magang dari minggu keenam hingga minggu kesepuluh dapat dilihat di Tabel 3.2.

Tabel 3.2 Tabel Timeline Aktivitas Kerja Magang Minggu Keenam hingga Minggu Kesepuluh

No	Aktivitas	Minggu ke-				
		6	7	8	9	10
12	Melakukan *update* data di tabel *quotation_items* di *MySQL*					
13	Menambahkan nama *sales* baru di *commission spreadsheets*					
14	Optimisasi kode *pricing engine* lem *data dump*					
15	Melakukan *update* data GOBIZ lanjutan di *MySQL*					
16	Menambahkan kolom *discount* di *gobiz data pipeline*					
17	Investigasi masalah data *google click id entry*					
18	Implementasi solusi untuk masalah *overloaded import_range*					
19	Mengubah format *pricing engine* menjadi per bulan dan format *timestamp*					
20	Membuat *spreadsheet paper offset history data dump*					
21	Membuat *sales-ops-npd dashboard* bersama *stakeholders* di *Google Spreadsheets*					

3.3.12. Melakukan *Update* Data di Tabel Quotation_Items di MySQL

Pada bagian ini, *stakeholders* ingin melakukan perubahan data di kolom yang terdapat pada tabel *quotation_items*. Perubahan dapat dilakukan dengan menggunakan *update script* dari *MySQL*. Proses *update* dilakukan berdasarkan *id.*

3.3.13. Menambahkan Nama Sales Baru di Commission Spreadsheets

Pada *request* ini, nama *sales* baru dapat langsung diregistrasikan dengan mengisi *spreadsheet* yang berisikan nama *sales* dan informasi-informasi pendukung lainnya.

3.3.14. Optimisasi Kode Pricing Engine Lem Data Dump

Kode untuk melakukan proses ETL terhadap data JSON yang berisikan informasi *pricing engine* – lem telah menjalankan tugasnya sebagaimana mestinya. Namun, mengingat data yang cukup besar dan *processing time* yang masih bisa dipersingkat. Kode ETL tersebut masih bisa dioptimisasi.

```
62    for month in defined_month:
63        month_kebab = month
64        month_snake = month.replace("-","_")
65
66        e7t_monthly = PythonOperator
          (task_id="calc_history_lem_e7t_spsheet_filter_month"+month_snake,
          python_callable=calc_history_lem_e7t_spsheet_filter_month.main,
          op_kwargs={'selected_time': month_kebab})
67
68        cl_monthly = PythonOperator( task_id="clear_load_"+month_snake,
          python_callable=clear_load.main, op_kwargs={ 'pipeline_name':
          'pricing_spsheet_lem', 'selected_time':month_kebab})
69
70        # defining dependency # equal to e.set_downstream(t), t.set_downstream
          (l)
71        e6 >> e7t_monthly >> cl_monthly
72
```

Gambar 3.11 Optimisasi Kode *Pricing Engine* – Lem

3.3.15. Melakukan Update Data GOBIZ Lanjutan di MySQL

Pada bagian ini, *stakeholders* ingin melakukan perubahan data di kolom *name* yang terdapat pada tabel *categories*. Perubahan dapat dilakukan dengan menggunakan *update script* dari *MySQL*. Proses *update* dilakukan berdasarkan *id.*

id	name	slug	master_category_id	createdAt	updatedAt
1	Paket Higienis GoFood	produk-higienis-gofood	9	11/17/2020 16:25	11/17/2020 16:25
2	Produk Promosi	produk-promosi	NULL	11/17/2020 16:25	11/17/2020 16:25
3	Kemasan Makanan	kemasan-makanan	6	11/17/2020 16:25	11/17/2020 16:25
4	Paket Hemat	paket-hemat	NULL	2/26/2021 20:14	2/26/2021 20:14

Gambar 3.12 Struktur dan Data Tabel *categories* di database GOBIZ

3.3.16. Menambahkan Kolom Discount di GOBIZ Data Pipeline

Output dari hasil transformasi data GOBIZ adalah sebuah file JSON yang didalamnya mengandung data per hari yang telah ditransformasi sesuai dengan format yang diterima GOBIZ. Hal ini membuat proses transformasi yang dilakukan berbeda dengan data *output spreadsheet* tradisional. Proses penambahan kolom *discount* dapat dilihat di Gambar 3.13.

```
98          for index, row in sub_df.iterrows():
99              product = {
100                 'product_id' : row['product_id_transitem'],
101                 'product_sku_name' : row['product_name_transitem'],
102                 'product_category' : row['name_category'],
103                 'product_subcategory' : "",
104                 'product_brand_name' : row['vendor_transitem'],
105                 'product_type' : "Non-Digital", #hard coded
106                 'product_price' : row['price_transitem'],
107                 'product_order_qty' : row['quantity_transitem'],
108                 'product_discount_price' : row['discount_price_transitem'],
109             }
110             grouped_df['products'].append(product)
```

Gambar 3. 13 Penambahan Kolom *Discount* di GOBIZ Data Pipeline

3.3.17. Investigasi Masalah Data Google Click Id Entry

Pada kasus ini, data *google click id entry* merupakan data yang diambil dari pengunjung ke situs PT. XYZ menggunakan fitur yang disediakan *Google*. Masalah yang terjadi adalah munculnya *inconsistency* dalam data yang menyebabkan *flow* dari pembentukan status sebagai pengunjung sampai ke membuat kuotasi menjadi ternganggu. Setelah diinvestigasi, masalah disebabkan oleh perilaku *sales team* dalam menangani suatu kuotasi sehingga akan dibuat SOP baru yang berada di luar ranah tim Data untuk menyelesaikan masalah ini.

3.3.18. Implementasi Solusi untuk Masalah Overloaded Import_Range

Pada PT. XYZ, data *dump* biasanya diakses menggunakan fungsi *import_range*, dimana pengguna bisa memanggil data secara langsung ke *spreadsheet* mereka bekerja. Akan tetapi, timbul kata *error* saat *import_range* digunakan untuk memanggil data yang jumlahnya relatif besar. Setelah dilakukan investigasi, masalah terjadi karena *import_range* mengalami *overloaded*. Solusi yang dapat diterapkan adalah menggunakan *import_range* hanya untuk memanggil 1 kolom pada suatu waktu sehingga tidak akan menyebabkan *overload.*

3.3.19. Mengubah Format Pricing Engine Menjadi per Bulan dan Format Timestamp

Hasil transformasi dari *pricing engine* dapat dibagi-bagi menjadi data per bulan. Hal ini dilakukan untuk meringankan *processing power* karena data *push* dilakukan menggunakan sistem *batch* atau per bulan dibandingkan dengan secara keseluruhan. Optimisasi dilakukan dengan mendefinisikan pembagian dengan format "YY-MM" yang dapat dilihat di Gambar 3.14. Perubahan format *timestamp* dilakukan dengan mengubah kode pada bagian *transformasi* ke format "YYYY-MM-DD hh:mm:ss".

```
35  # daily at 18 UTC or 01:00 AM GMT+7
36  with DAG( dag_id,
37            start_date=datetime( 2020, 3, 21 ),
38            schedule_interval="0 18 * * *",
39            default_args=DAG_DEFAULT_FILES,
40            catchup=False ) as dag:
41
42      # defining task
43      e6 = PythonOperator( task_id="calc_history_lem_e6",
44                           python_callable=calc_history_lem_e6.main )
45
46      # define task for loading to each month
47      defined_month=[
48          "20-12",
49          "20-11",
50          "20-10",
51          "20-09",
52          "20-08",
53          "20-07",
54          "20-06",
55          "20-05",
56          "20-04",
57          "20-03",
58          "20-02",
59          "20-01",
60      ]
```

Gambar 3. 14 Optimisasi Data *Push Pricing Engine*

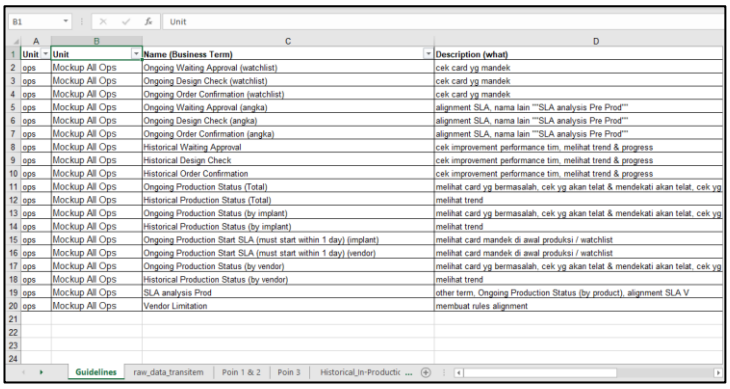
3.3.20. Membuat Spreadsheet Paper Offset History Data Dump

Data dari *paper offset history* juga disimpan dalam bentuk format JSON. Kode *extract* mengambil informasi file JSON yang menyimpan data *paper offset history*. Dikarenakan tipe file yang berbeda, proses *extract* berbeda dengan cara konvensional biasanya. Setelah melalui proses ekstraksi,

data akan dilakukan transformasi sesuai dengan kebutuhan stakeholders dan akan di-*push* ke spreadsheet *paper offset history dump* untuk digunakan.

3.3.21. Membuat Sales-Ops-Npd Dashboard Bersama Stakeholders di Google Spreadsheets

Sales-Ops-Npd *Dashboard* merupakan salah satu projek *stakeholders* yang bertujuan untuk menyatukan *dashboard-dashboard* tidak terintegrasi yang biasanya dipakai oleh divisi Sales-Ops-Npd. Peran *data engineer intern*, dalam hal ini adalah membantu *stakeholder* terkait dalam membuat *formula* yang efisien dan *scalable*. Ketiga *spreadsheet* yang memiliki kumpulan *dashboard-dashboard* dapat digunakan oleh divisi terkait untuk kebutuhan *decision making* yang dapat dilihat di Gambar 3.15.



	A	B	C	D
1	Unit	Unit	Name (Business Term)	Description (what)
2	ops	Mockup All Ops	Ongoing Waiting Approval (watchlist)	cek card yg mandek
3	ops	Mockup All Ops	Ongoing Design Check (watchlist)	cek card yg mandek
4	ops	Mockup All Ops	Ongoing Order Confirmation (watchlist)	cek card yg mandek
5	ops	Mockup All Ops	Ongoing Waiting Approval (angka)	alignment SLA, nama lain ""SLA analysis Pre Prod""
6	ops	Mockup All Ops	Ongoing Design Check (angka)	alignment SLA, nama lain ""SLA analysis Pre Prod""
7	ops	Mockup All Ops	Ongoing Order Confirmation (angka)	alignment SLA, nama lain ""SLA analysis Pre Prod""
8	ops	Mockup All Ops	Historical Waiting Approval	cek improvement performance tim, melihat trend & progress
9	ops	Mockup All Ops	Historical Design Check	cek improvement performance tim, melihat trend & progress
10	ops	Mockup All Ops	Historical Order Confirmation	cek improvement performance tim, melihat trend & progress
11	ops	Mockup All Ops	Ongoing Production Status (Total)	melihat card yg bermasalah, cek yg akan telat & mendekati akan telat, cek yg
12	ops	Mockup All Ops	Historical Production Status (Total)	melihat trend
13	ops	Mockup All Ops	Ongoing Production Status (by implant)	melihat card yg bermasalah, cek yg akan telat & mendekati akan telat, cek yg
14	ops	Mockup All Ops	Historical Production Status (by implant)	melihat trend
15	ops	Mockup All Ops	Ongoing Production Start SLA (must start within 1 day) (implant)	melihat card mandek di awal produksi / watchlist
16	ops	Mockup All Ops	Ongoing Production Start SLA (must start within 1 day) (vendor)	melihat card mandek di awal produksi / watchlist
17	ops	Mockup All Ops	Ongoing Production Status (by vendor)	melihat card yg bermasalah, cek yg akan telat & mendekati akan telat, cek yg
18	ops	Mockup All Ops	Historical Production Status (by vendor)	melihat trend
19	ops	Mockup All Ops	SLA analysis Prod	other term, Ongoing Production Status (by product), alignment SLA V
20	ops	Mockup All Ops	Vendor Limitation	membuat rules alignment
21				
22				
23				
24				

Guidelines | raw_data_transitem | Poin 1 & 2 | Poin 3 | Historical_In-Productic ...

Gambar 3.15 *Operation Dashboard Spreadsheet*

Timeline aktivitas kerja magang dari minggu kesebelas hingga minggu kelima belas dapat dilihat di Tabel 3.3.

Tabel 3.3 Tabel Timeline Aktivitas Kerja Magang Minggu Kesebelas hingga Minggu Kelima Belas

No	Aktivitas	Minggu ke-				
		11	12	13	14	15
22	Melakukan *update* dokumentasi					
23	Mengganti sumber data *NPD SLA Pivot Table* ke *at_rfq* di *spreadsheet*					
24	Melakukan investigasi dan perbaikan terhadap *spreadsheet* yang sumber datanya *error*.					
25	Melakukan *sync* nama produk di PT. XYZ					
26	Menambahkan kolom *at_negotiation* ke *quoted data dump*					
27	Melakukan *refactor Manager Commercial Dashboard*					
28	Implementasi *array formula* dan finalisasi *Sales-Ops-Npd Dashboard*					
29	Membuat *vendor retention data dump*					
30	Menambahkan kolom *price_transitem* ke *quoted data dump*					
31	Membuat sistem *autofill* di *Tokopedia Dashboard*					

3.3.22. Melakukan Update Dokumentasi

Dalam tim Data di PT. XYZ, terdapat 3 (tiga) jenis dokumentasi yaitu *data definitions*, *data dependencies*, dan *data state*. *Data definitions* merupakan dokumentasi yang mencatat arti dari kolom suatu tabel. Peran dari dokumentasi ini adalah memberikan konteks ke si pengguna. *Data dependencies* menunjukkan fitur-fitur yang bergantung ke suatu kolom tabel atau data tertentu sehingga menjadi faktor pertimbangan apabila suatu kolom atau data akan dilakukan perubahan. *Data state* menunjukkan tingkat kualitas dari suatu data.

3.3.23. Mengganti Sumber Data NPD SLA Pivot Table ke At_rfq di Spreadsheet

Pada kasus ini, *stakeholders* menilai bahwa sumber data *NPD SLA Pivot Table* yang lama yaitu *at_quoted* sudah tidak lagi sesuai dengan kebutuhan. Pergantian variabel ke *at_rfq* dilakukan dengan menggunakan UI yang disediakan oleh *Google Spreadsheets*.

3.3.24. Melakukan Investigasi dan Perbaikan Terhadap Spreadsheet yang Sumber Datanya Error

Penggunaan *spreadsheets* yang dilakukan oleh karyawan PT. XYZ memberikan fleksibilitas yang luar biasa, akan tetapi, salah satu kelemahan yang ada adalah lemahnya integrasi atau standarisasi sehingga menyebabkan *formula* yang digunakan untuk memanggil suatu sumber data masih menggunakan versi lama yang sudah tidak mampu lagi menanggung beban sumber data yang semakin membesar. Hasil dari investigasi menunjukkan tanda-tanda *overload* sehingga solusi yang dilakukan adalah semua *formula* diubah ke versi yang lebih efisien dan *scalable.*

3.3.25. Melakukan Sync Nama Produk di PT. XYZ

Pada kasus ini, sejumlah data nama produk yang berada di tabel *transaction_items* di *MySQL* memiliki perbedaan dengan standar nama produk yang disimpan di *master products spreadsheet*. Hal ini menyebabkan kebingungan dan data yang kurang berkualitas diantara pengguna di PT.

XYZ. Dalam menyelesaikan masalah ini, *master products spreadsheet* digunakan sebagai acuan nama produk yang benar dengan menggunakan informasi di kolom *updated name* sebagai referensi yang dapat dilihat di Gambar 3.16.

id	Name	Updated Name
235	Agenda	
289	Agenda Hardcover Fancy	
290	Agenda Sintetis Fancy	
282	Alumunium Foil Bag Doff	
247	Amplop	
307	Bagasse Lunch Box	
129	Biodegradable Plastic	
271	Biodegradable Poly Mailer	
	Biodegradable Shopping Bag	
328	Bluetooth Speaker	
390	Botol Plastik Clio	
113	Box Kardus	Box Kardus Standar
239	Box Kardus Full Color (Duplex)	Box Kardus Duplex Standar
112	Box Kardus Polos	Part of plain product
187	Box Kentang	
185	Box Nasi	
238	Box Pengiriman	Box Kardus Mailer
240	Box Pengiriman Full Color (Duplex)	Box Kardus Duplex Mailer
106		Part of spec
108	Box Product Classic	Box Product Classic
109	Box Product Two Piece	Box Product Two Piece
4	Brosur	
308	Bubble Mailer	Bubble Mailer Printing
308	Bubble Mailer Sablon	

Gambar 3.16 *Master Product Spreadsheet*

Proses *sync* dilakukan dengan mengubah data nama produk di *MySQL* yang tidak tepat menggunakan *update script*.

3.3.26. Menambahkan Kolom At_negotiation ke Quoted Data Dump

Kolom *at_negotiation* dapat ditambahkan dengan memasukkan nama kolom *at_negotiation* ke daftar kolom yang akan di-*push* ke *spreadsheet quoted data dump*. Dikarenakan data sudah ada di *transformed* data, maka hanya tinggal perlu memodifikasi filter kolom saja.

3.3.27. Melakukan Refactor Manager Commercial Dashboard

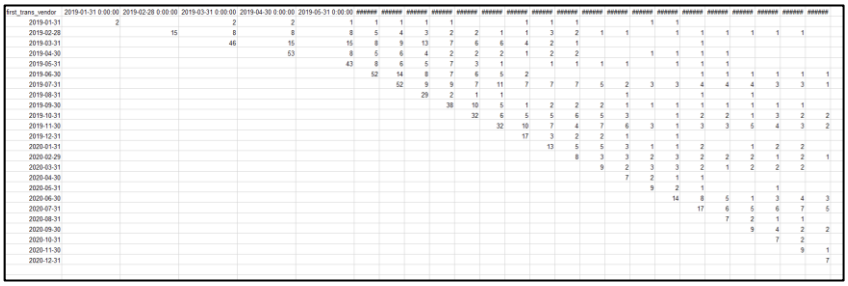
Alasan dilakukannya *refactor* pada *Manager Commercial Dashboard* adalah dikarenakan *spreadsheet* tersebut sebentar lagi akan mencapai batas jumlah *cell* yang ditetapkan oleh *Google Spreadsheets* yaitu sebesar 5 (lima) juta *cells*. Proses *refactor* dipertimbangkan dengan menggabungkan jenis *dashboard* yang biasanya dipakai secara bersamaan ke *spreadsheet* lain.

3.3.28. Implementasi Array Formula dan Finalisasi Sales-Ops-Npd Dashboard

Sales-Ops-Npd dashboard spreadsheet yang telah dibuat akan diimplementasikan dengan penggunaan *array formula*. Hal ini dilakukan agar komputasi dapat tetap *up-to-date* dikarenakan fungsi *array formula* yang secara otomatis mengakomodir data-data baru. Selain itu, *dashboard-dashboard* ini juga sedang diminta masukan dan *feedbacknya* dari divisi yang akan menggunakannya. Berdasarkan masukan itu, maka perubahan terhadap *dashboard* akan terus dilakukan sampai periode waktu yang telah ditentukan.

3.3.29. Membuat Vendor Retention Data Dump

Berdasarkan *request* dari *stakeholders, vendor retention data dump* dibuat untuk menunjukkan frekuensi seringnya suatu *vendor* digunakan dalam percetakan produk. *Vendor retention data dump* dapat dilihat di Gambar 3.17.



Gambar 3. 17 Data *Vendor Retention Data Dump Periode 2019 sampai 2020*

3.3.30. Menambahkan Kolom Price_transitem ke Quoted Data Dump

Kolom *at_price* dapat ditambahkan dengan memasukkan nama kolom *at_price* ke daftar kolom yang akan di-*push* ke *spreadsheet quoted data dump*. Dikarenakan data sudah ada di *transformed* data, maka hanya tinggal perlu memodifikasi filter kolom saja.

3.3.31. Membuat Sistem Autofill di Tokopedia Dashboard

Pada kasus ini, *formula* pada *Google Spreadsheets* yang masih menggunakan metode lama membuat *formula* tersebut tidak dapat secara otomatis mengikutsertakan data baru sehingga menyebabkan *delay* dalam perhitungan. Masalah ini dapat diselesaikan dengan melakukan implementasi *array formula* yang dapat secara otomatis mengikat data baru.

Timeline aktivitas kerja magang dari minggu keenam belas hingga minggu keduapuluh dapat dilihat di Tabel 3.4.

Tabel 3.4 Tabel Timeline Aktivitas Kerja Magang Minggu Keenam Belas hingga Minggu Keduapuluh

No	Aktivitas	Minggu ke-				
		16	17	18	19	20
32	Melakukan *populate* pada kolom *in_production_qty* di tabel *transaction_items*					
33	Melakukan *update* data vendor di *MySQL*					
34	Melakukan *sync* nama vendor di tabel *transaction_items* dan database *tjetak_barnacle*					
35	Melakukan optimisasi kode *vendor retention data dump*					
36	Melakukan optimisasi *formula autofill* di *Tokopedia Dashboard*					
37	Memperbaiki *grouped_duration* di *SLA NPD Pivot Table*					
38	Melakukan standarisasi nama vendor dan *id* vendor antara 3 (tiga) sumber data					
39	Melakukan kalkulasi *in-production SLA* di *transaction_items*					
40	Melakukan *clear-out* data AP *(Account Payable)*					
41	Melakukan *update* ke *commission calculation spreadsheet*					
42	Menambahkan kolom *customer_price_request* di *quoted data dump*					
43	Melakukan *revert* data terhadap nama produk yang ambigu					
44	Melakukan *status update* data konsumen *eyelovin* dari *cancel* menjadi *done*					

3.3.32. Melakukan Populate pada Kolom In_production_qty di Tabel Transaction_items

Pada kasus ini, *in_production_qty* merupakan kolom baru yang dibuat untuk merepresentasikan jumlah barang yang sedang diproduksi. Sebagai inisialisasi awal, *stakeholders* meminta *data engineer intern* untuk

menggunakan *qty* atau jumlah barang pada tiap *transaction_items* sebagai nilai yang digunakan untuk melakukan *populate* ke kolom terkait. Proses *populate* menggunakan *update syntax* yang terdapat di *MySQL*.

3.3.33. Melakukan Update Data Vendor di MySQL

Data engineer intern mendapatkan *request* dari *stakeholders* untuk melakukan *update* ke data vendor yang ada di *database* untuk memastikan data *up-to-date*. Setelah berkomunikasi dengan *stakeholders* terkait, informasi yang didapatkan kemudian digunakan untuk melakukan *update* ke data vendor yang ada di *database*. Proses *update* menggunakan *update syntax* yang ada di *MySQL*.

3.3.34. Melakukan Sync Nama Vendor di Tabel Transaction_items dan Database Tjetak_barnacle

Data vendor di *database tjetak_barnacle* berfungsi sebagai data *master* yang harus diikuti oleh semua kolom yang berhubungan dengan data vendor. Akan tetapi, saat investigasi dilakukan, banyak nama vendor di data *transaction_items* yang tidak sesuai dengan data di *database tjetak_barnacle*. Setelah ditelusuri lebih lanjut, hal ini disebabkan oleh proses *assign* vendor ke suatu *transaction_items* yang masih menggunakan *input* manual sehingga menyebabkan inkonsistensi nama vendor. Proses *sync* yang dilakukan adalah melakukan *update* terhadap data *transaction_items* agar mengikuti format *master*.

3.3.35. Melakukan Optimisasi Kode Vendor Retention Data Dump

Berdasarkan *feedback* yang diperoleh dari *stakeholders*, *vendor retention data dump* yang dibuat kurang akurat jika dibandingkan dengan kalkulasi manual. Setelah dieksplorasi dan didiskusikan secara bersamaan, masalah terletak pada kurangnya *filter* sehingga menyebabkan data yang seharusnya tidak termasuk, ikut terhitung kedalam perhitungan *retention*. Proses implementasi filter pada kode dapat dilihat pada Gambar 3.18.

```
120  # FILTER LIST
121  # name_status is done
122  def filter_done_only(df):
123      return df[ df['name_status'] == 'done']
124
125  # final_vendor_transitem is Exist
126  def filter_no_vendor(df):
127      return df[ df['final_vendor_transitem'] != '']
128
129  def transform(df):
130      df['transitem_is_valid'] = df.apply(lambda row: enrich_is_valid_sales(row,
         '','_transitem'), axis = 1)
131      return df
132
133  def enrich_is_valid_sales(row,prefix,suffix):
134      conditions = True
135      conditions = conditions and row[prefix+'status_id'+suffix] != 2 #-
         transaksi yg ada di "rfq"
136      conditions = conditions and row[prefix+'status_id'+suffix] != 5 #-
         transaksi yg ada di "cancel"
137      conditions = conditions and row[prefix+'status_id'+suffix] != 14 #
         transaksi yg ada di "cancelled" jangan dibaca, datanya berantakan, ga valid
138      conditions = conditions and row[prefix+'status_id'+suffix] != 15 #-
         "waiting_payment" juga jangan dibaca,
139      conditions = conditions and row[prefix+'status_id'+suffix] != 16 #-
         "failed" juga jangan dibaca,
140      conditions = conditions and row[prefix+'status_id'+suffix] != 17 #-
         "return" jangan dibaca juga
141      conditions = conditions and pd.isna(row[prefix+'deleted_at'+suffix]) #
         jangan tarik yg udah ke delete
142      return 'valid' if conditions else 'invalid'
```

Gambar 3. 18 Implementasi *Filter* pada *Vendor Retention Data Dump*

3.3.36. Melakukan Optimisasi Formula Autofill di Tokopedia Dashboard

Formula yang telah menggunakan *array formula* memastikan bahwa seluruh hasil komputasi akan bersifat *up-to-date* dikarenakan data baru akan selalu secara otomatis dimasukkan ke dalam perhitungan. Namun, *formula* komputasi masih bisa dibuat lebih efisien lagi dengan mengurangi

penggunaan fungsi *If-ElseIf, Or, And* secara berlebihan. Pada kasus ini, hal yang ingin dicapai adalah *formula* simpel yang dapat mencapai tujuan yang sama.

3.3.37. Memperbaiki Grouped_duration di SLA NPD Pivot Table

Grouped duration yang berada di *SLA NPD Pivot Table* mengalami malfungsi, batas awal yang seharusnya 0 (nol) tidak muncul. Akan tetapi, batas awal yang muncul langsung dimulai dari angka 1 (satu). Masalah ini dapat diatasi dengan melakukan *re-apply* pada *grouped duration* dengan nilai yang benar.

3.3.38. Melakukan Standarisasi Nama Vendor dan Id Vendor Antara 3 (tiga) Sumber Data

Proses standarisasi nama vendor dan *id* vendor yang dilakukan adalah antara data AP *(Account Payable)* milik divisi keuangan, dan data di tabel *transaction_items* dan data di *database tjetak_barnacle*. Berdasarkan *stakeholders*, acuan yang akan digunakan adalah data yang sudah dilakukan *sync* sebelumnya yaitu data di *database tjetak_barnacle*. Tugas *data engineer intern* disini adalah melakukan *highlight* terhadap data milik keuangan yang tidak sesuai agar dapat diubah dan distandarisasi.

3.3.39. Melakukan Kalkulasi In-production SLA di Transaction_items

Pada kasus ini, *stakeholders* menemukan adanya data SLA *in-production* yang isinya *blank*. Jika dipikirkan sesuai SOP yang berlaku, hal ini seharusnya tidak mungkin terjadi. Akan tetapi, isu mengenai ini akan diinvestigasi oleh pihak lain, sedangkan tugas *data engineer intern* disini adalah melakukan kalkulasi secara manual agar data *in-production* SLA dapat muncul. Kalkulasi ini dapat dengan melakukan perbandingan tanggal orderan masuk, tanggal orderan memasuki SLA *design-check*, dan tanggal orderan memasuki SLA *waiting-confirmation*.

3.3.40. Melakukan Clear-out Data AP (Account Payable)

Data AP *(Account Payable)* milik keuangan akan dilakukan proses *clear-out* dikarenakan rencananya data AP tersebut akan dimigrasi ke *database* untuk mendukung aplikasi atau fitur lain. Akan tetapi, dikarenakan selama ini pencatatan dilakukan di *Google Spreadsheet* yang tidak terintegrasi dan memiliki cara pencatatan yang unik. Maka faktor ini menambahkan kesulitan dalam menerjemahkan data AP *(Account Payable)* ke format yang diterima oleh *database*. Namun, berkat koordinasi dengan *stakeholders* yang kuat dan jelas, data AP dapat di *clearout* dan kemudian siap dimasukkan ke *databases* menggunakan *insert syntax* di *MySQL*.

3.3.41. Melakukan Update ke Commission Calculation Spreadsheet

Pada kasus ini, terdapat permintaan dari *stakeholders* untuk melakukan perubahan data *target limit* di kategori *new account*. Perubahan dapat dilakukan secara langsung dengan merubah *value* yang ada di *sales commission spreadsheet*.

3.3.42. Menambahkan Kolom Customer_price_request di Quoted Data Dump

Kolom *customer_price_request* dapat ditambahkan dengan memasukkan nama kolom *at_price* ke daftar kolom yang akan di-*push* ke *spreadsheet quoted data dump*. Dikarenakan data sudah ada di *transformed* data, maka hanya tinggal perlu memodifikasi filter kolom saja.

3.3.43. Melakukan Revert Data Terhadap Nama Produk yang Ambigu

Nama produk yang berada di data tabel *transaction_items* sebelumnya telah dirubah untuk mengikuti standar nama produk yang ada di *master products spreadsheet*. Akan tetapi, setelah ditelusuri, terdapat beberapa nama produk di *master product* yang membuat definisinya menjadi ambigu dikarenakan mirip dengan nama produk lain. Hal ini dapat menyebabkan kesalahan fatal apabila pengguna data tidak tau akan isu terkait, sehingga diputuskan untuk melakukan *revert* balik sejumlah data nama produk ke nama produk yang lama. Hal ini dilakukan dengan menggunakan *update syntax* di *MySQL*.

3.3.44. Melakukan Status Update Data Konsumen Eyelovin dari Cancel Menjadi Done

Berdasarkan hasil diskusi dengan *stakeholders*, terdapat beberapa data konsumen bernama *eyelovin* yang status transaksinya berada di status *cancel*. Padahal, transaksi tersebut telah diproses dan dibayarkan oleh konsumen terkait, sehingga harusnya status yang benar adalah *done*. Proses perubahan dilakukan dengan mengidentifikasi data yang bermasalah dan melakukan *update* menggunakan *update syntax* di *MySQL*.

Timeline aktivitas kerja magang dari minggu keduapuluh satu hingga minggu keduapuluh lima dapat dilihat di Tabel 3.5.

Tabel 3.5 Tabel Timeline Aktivitas Kerja Magang Minggu Keduapuluh Satu hingga Minggu Keduapuluh Lima

No	Aktivitas	Minggu ke-				
		21	22	23	24	25
45	Melakukan perbaikan terhadap *formula* yang error di *spreadsheet OKR NPD*					
46	Membuat *quoted data dump* versi tahun 2018 dan 2019					
47	Menambahkan kolom *address* ke *shipping data*					
48	Melakukan migrasi *quoted* dan *transitem data dump* ke *Google BigQuery*					
49	Mengubah *LoD* pada tasks					
50	Melakukan perbaikan pada *conflicting code* pada bagian e21 di *dags transitem and NOD*					
51	Menambahkan kolom *vendor_assignment* di *transitem data dump*					
52	Melakukan inisialisasi pada kode agar kolom baru dapat muncul di *dump BigQuery*					

53	Melakukan *clearout* data *rank* dan *status* pada tabel *vendor choices* di *MySQL*					
54	Menambahkan *transaction_cancellation_reason* ke kode di *gobiz daily push*					
55	Melakukan *clearout* data nama vendor pada tabel *transaction_items*					
56	Melakukan *development* kode untuk melakukan data *push* ke *MoEngage*					
57	Melakukan investigasi terhadap *AR (Account Receivable) spreadsheet error*					
58	Melakukan investigasi data *at_[waitingapproval/designcheck/etc]* yang hilang					

3.3.45. Melakukan Perbaikan Terhadap Formula yang Error di Spreadsheet OKR NPD

Formula yang error pada *spreadsheet OKR NPD* disebabkan oleh *typo* yang muncul pada tengah-tengah *formula*. Setelah dilakukan pengecekan menggunakan *version history* dari *spreadsheet* tersebut, ternyata *typo* muncul dari ketidaksengajaan salah satu pengguna *spreadsheet* tersebut. Solusi yang dilakukan adalah menggunakan fungsi *protected cell* sehingga membatasi jumlah orang yang bisa melakukan perubahan pada *cell Formula* terkait.

3.3.46. Membuat Quoted Data Dump Versi Tahun 2018 dan 2019

Berdasarkan penjelasan *stakeholders*, terkadang ada saatnya dimana pengguna data *quoted data dump* ingin melakukan analisis ke data lampau seperti tahun 2019 ataupun sampai 2018. Untungnya, proses *transformasi* data yang dilakukan di *Airflow* mencakup semua periode data sehingga

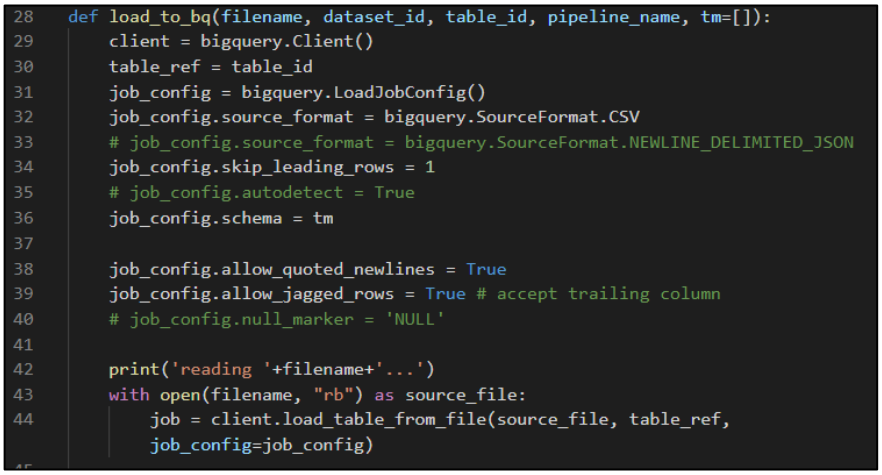
request ini dapat dengan mudah diselesaikan dengan melakukan modifikasi di *defined month* dan *load code*.

3.3.47. Menambahkan Kolom Address ke Shipping Data

Kolom *address* dapat ditambahkan dengan memasukkan nama kolom *address* ke daftar kolom yang akan di-*push* ke *shipping data*. Dikarenakan data sudah ada di *transformed* data, maka hanya tinggal perlu memodifikasi filter kolom saja.

3.3.48. Melakukan Migrasi Quoted dan Transitem Data Dump ke Google BigQuery

Pada kasus ini, Data *quoted* dan *transitem* yang telah melalui tahap *transformation* akan di-*load* ke *Google BigQuery*. Perbedaan utama dari melakukan *load* ke *spreadsheets* dan *load* ke *BigQuery* adalah jika di *spreadsheets* maka tidak perlu adanya inisalisasi kolom dengan tipe data terlebih dahulu. Flexibilitas ini datang dengan batasan maksimal 5 (lima) juta *cells* sedangkan *BigQuery* memiliki beberapa langkah tambahan tapi memang didesain untuk menampung data dengan jumlah besar atau *big data*. Konfigurasi *load* data ke *BigQuery* dapat dilihat di Gambar 3.19.



```
def load_to_bq(filename, dataset_id, table_id, pipeline_name, tm=[]):
    client = bigquery.Client()
    table_ref = table_id
    job_config = bigquery.LoadJobConfig()
    job_config.source_format = bigquery.SourceFormat.CSV
    # job_config.source_format = bigquery.SourceFormat.NEWLINE_DELIMITED_JSON
    job_config.skip_leading_rows = 1
    # job_config.autodetect = True
    job_config.schema = tm

    job_config.allow_quoted_newlines = True
    job_config.allow_jagged_rows = True # accept trailing column
    # job_config.null_marker = 'NULL'

    print('reading '+filename+'...')
    with open(filename, "rb") as source_file:
        job = client.load_table_from_file(source_file, table_ref,
        job_config=job_config)
```

Gambar 3. 19 Konfigurasi *Load* Data ke *BigQuery*

3.3.49. Mengubah LoD pada Tasks

Berdasarkan hasil diskusi dengan *data lead*, ternyata telah terjadi kesalahan dalam tata cara dan *flow* penulisan kode. Salah satu *task* yaitu *transitem_e4.py* seharus memiliki *LoD (Level of Detail)* di tahap *transaction_items.* Namun, akibat proses *join* data yang kurang tepat, *LoD* data berubah menjadi *transaction*, dimana *LoD* ini berada di satu level diatas *transaction_items* sehingga menyebabkan banyak data yang error. Solusi dari masalah ini adalah melakukan evaluasi kembali terhadap kode di *transitem_e4.py* dan melakukan perbaikan agar *LoD* kembali ke seperti semula.

3.3.50. Melakukan Perbaikan pada Conflicting Code pada Bagian e21 di Dags Transitem and NOD

Task dengan judul *file history_e21.py* memiliki tugas untuk melakukan ekstraksi data dari tabel *history* yang mengandung seluruh jejak perubahan *quotation* maupun *transaction_items*. Setelah berdiskusi dengan *data lead*, letak kesalahan terletak pada tidak adanya *filter* jenis *quotation* atau *transaction_items*. *Filter* seharusnya dipergunakan untuk hanya mengambil data dengan tipe *transaction_items* untuk menghasilkan hasil akhir yang akurat.

3.3.51. Menambahkan Kolom Vendor_assignment di Transitem Data Dump

Kolom *vendor_assignment* dapat ditambahkan dengan memasukkan nama kolom *vendor_assignment* ke daftar kolom yang akan di-*push* ke *transitem data dump*. Dikarenakan data sudah ada di *transformed* data, maka hanya tinggal perlu memodifikasi filter kolom saja.

3.3.52. Melakukan Inisialisasi pada Kode Agar Kolom Baru Dapat Muncul di Dump BigQuery

Proses inisialisasi kode agar *load* data ke *BigQuery* dapat berhasil adalah dengan memastikan nama kolom dan tipe kolom yang ditulis di kode sama dengan informasi yang ada di tabel *BigQuery*. Inisialisasi kode ini berfungsi sebagai semacam *identifier* ketika data akan di-*load*.

3.3.53. Melakukan Clearout Data Rank dan Status pada Tabel Vendor Choices di MySQL

Pada kasus ini, data *rank* dan *status* pada tabel *vendor choices* akan di *clearout* sesuai dengan instruksi dari *stakeholders*. Berdasarkan informasi mengenai *rank* dan *status* yang benar dan diberikan oleh *stakeholders*, maka proses *clearout* dapat dilakukan dengan melakukan *update* ke data *rank* dan *status* di tabel *vendor choices* dengan menggunakan *update syntax* di *MySQL*.

3.3.54. Menambahkan Transaction_cancellation_reason ke Kode di Gobiz Daily Push

Output dari hasil transformasi data GOBIZ adalah sebuah file JSON yang didalamnya mengandung data per hari yang telah ditransformasi sesuai dengan format yang diterima GOBIZ. Hal ini membuat proses transformasi yang dilakukan berbeda dengan data output spreadsheet tradisional. Proses penambahan kolom *transaction_cancellation_reason* dapat dilihat di Gambar 3.20.

```
174     # reason
175     result_df['transaction_cancellation_reason'] = df.apply(lambda row: (row
        ['cancel_reason_transaction'] if row['status_transaction'] =='cancelled'
        else ""), axis=1)
```

Gambar 3. 20 Penambahan Kolom *transaction_cancellation_reason* di GOBIZ Data

3.3.55. Melakukan Clearout Data Nama Vendor pada Tabel Transaction_items

Dalam data *transaction_items*, masih terdapat data-data yang tidak memiliki nama vendor yang sesuai dengan *master* data. Sebagai *data engineer intern*, tugas yang dilakukan pada kasus ini adalah mengumpulkan data-data yang nama vendornya tidak jelas. Lalu, mengkomunikasikan data tersebut langsung ke *stakeholders* yang bersangkutan. Setelah *stakeholders* terkait memberikan balasan, maka perubahan yang akan dilakukan di *MySQL* dapat dilakukan menggunakan *update syntax.*

3.3.56. Melakukan Development Kode untuk Melakukan Data Push ke MoEngage

MoEngage merupakan sebuah *marketing platform* yang dapat digunakan untuk memberikan penawaran-penawaran dengan konfigurasi yang canggih berdasarkan dengan jenis kustomer yang sesuai dengan kriteria tertentu. Agar *platform* ini dapat menjalankan tugasnya, *MoEngage* memerlukan data *customer* beserta dengan informasi pendukung seperti jumlah frekuensi seorang konsumen pernah bertransaksi di PT. XYZ dan sebagainya. Setelah data yang dibutuhkan di ekstraksi, maka proses transformasi yang dilakukan adalah membuat format data menjadi *JSON Compatible*. Setelah JSON yang berisikan data telah siap, maka data dapat mulai di *push* ke *MoEngage* menggunakan API yang telah disediakan.

3.3.57. Melakukan Investigasi Terhadap AR (Account Receivable) Spreadsheet Error

Berdasarkan hasil diskusi bersama dengan *stakeholders*, masalah *error* yang muncul pada kalkulasi menggunakan *formula* adalah karena beban *computing* yang terlalu besar. Asumsi ini didukung dengan lamanya proses *loading spreadsheet* ketika seseorang ingin mengakses *AR spreadsheet* tersebut. Dikarenakan hal tersebut merupakan limitasi dari *Google Spreadsheets* itu sendiri, maka solusi yang akan dilakukan adalah memisahkan beberapa komponen yang ada pada *spreadsheet* tersebut ke *spreadsheet* lain guna mengurangi beban *computing* pada *AR spreadsheet*.

3.3.58. Melakukan Investigasi Data At_[waitingapproval/designcheck/..] yang Hilang

Stakeholders melaporkan isu bahwa terdapat data *at_[waitingapproval/designcheck/etc]* yang tiba-tiba hilang. Data ini merupakan data yang di *generate* menggunakan *history* data. Berdasarkan informasi ini, maka investigasi ke dalam kode pun dilakukan. Setelah berdiskusi dengan *data lead*, masalah yang menyebabkan hal ini terjadi adalah proses *extract* data yang kurang tepat sehingga menyebabkan sebagian data menjadi hilang.

3.4. Kendala yang Dihadapi

Dalam pelaksanaan kerja magang di PT. XYZ, terdapat beberapa kendala yang dihadapi yaitu sebagai berikut:

1. Kesulitan dalam mengikuti kultur kerja PT. XYZ yang *fast paced* pada awal kerja magang dikarenakan oleh kurangnya penguasaan mengenai hal-hal seperti penggunaan *scheduler software Airflow*, penggunaan *Python* untuk *advanced data manipulation*, dan penggunaan Git. Hal ini menyebabkan sulitnya mencapai ekspektasi yang dimiliki oleh *data lead* mengenai kualitas dan waktu penyelesaian suatu pekerjaan.
2. Sistem kerja *WFH (Work From Home)* membuat semua komunikasi harus dilakukan secara daring sehingga meningkatkan kemungkinan terjadinya kesalahpahaman saat berdiskusi.
3. Minimnya kesempatan berkomunikasi dengan *data lead* diluar *meeting* untuk meminta masukan atau bantuan dikarenakan oleh waktu respons yang sangat lambat. Hal ini dapat menciptakan *blocker* di pekerjaan yang membutuhkan intervensi langsung dari *data lead* dalam penyelesaiannya.

3.5. Solusi atas Kendala

Beberapa solusi yang dilakukan oleh mahasiswa untuk masalah yang dihadapi saat kerja magang di PT. XYZ adalah sebagai berikut:

1. Mempelajari *software Airflow*, *advanced data manipulation* menggunakan *Python*, dan fundamental Git dengan memanfaatkan *online resources* yang beranekaragam dan gratis. Selain itu juga, mahasiswa membiasakan diri

untuk selalu bertanya apabila *data lead* sedang menjelaskan mengenai topik diatas.

2. Melakukan komunikasi yang bersifat penting menggunakan *Conference Call* seperti *Zoom* atau *Google Meet* dikarenakan aplikasi-aplikasi ini memiliki *track record* yang bagus dan dapat diandalkan pada setiap saat.
3. Menyusun daftar pertanyaan dan topik yang ingin dibicarakan untuk kemudian dibahas saat sedang memiliki *meeting* dengan *data lead* seperti *daily standup meeting*. Proses pencatatan dilakukan di *code editor Visual Studio Code* dan disimpan pada 1 *file* khusus untuk efisiensi waktu dikarenakan mayoritas topik yang ingin dibicarakan merupakan masalah yang timbul saat proses *coding* menggunakan *code editor* yang sama.